Orientación a Objetos 1	**Carrera/ Plan:** *(Dejar lo que corresponda)* *Licenciatura en Informática* Plan 2015/Plan 2012/Plan 2003-07 *Licenciatura en Sistemas* Plan 2015/Plan 2012/Plan 2003-07 *Analista Programador Universitario* Plan 2015/Plan 2007 *Analista en TIC* Plan 2017
Año 2019	**Año:** 2019 **Régimen de Cursada:** *Semestral* **Carácter (Obligatoria/Optativa):** Obligatoria **Correlativas**: Taller de Programación **Profesor/es:** Gustavo Rossi, Roxana Giandini, Alicia Diaz **Hs. semanales**: 6 hrs

FUNDAMENTACIÓN

La Programación Orientada a Objetos (POO) es fundamental en la formación del profesional informático. La misma aporta técnicas de programación que combinan la abstracción, modularización, encapsulamiento junto con el polimorfismo y herencia, promoviendo así una forma de programación basada en la descomposición adecuada en objetos de manera de favorecer la modificabilidad, mantenimiento y reuso. Aspectos que son rectores en el desarrollo de un programa OO.

OBJETIVOS GENERALES

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

COMPETENCIAS

- CGS6- Capacidad para interpretar la evolución de la Informática con una visión de las tendencias tecnológicas futuras.
- CGT1- Identificar, formular y resolver problemas de Informática.
- CGT2- Concebir, diseñar y desarrollar proyectos de Informática.
- CGT5- Utilizar de manera efectiva las técnicas y herramientas de aplicación de la Informática.
- CGT8 Capacidad de interpretación y resolución de problemas multidisciplinarios, desde los conocimientos de la disciplina informática.
- LI - CE4 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real, especificación formal de los mismos, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software/sistemas de información que se ejecuten sobre equipos de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.
- LS - CE1 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real. Especificación formal, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software que se ejecuten sobre sistemas de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.

CONTENIDOS MINIMOS (de acuerdo al Plan de Estudios)

- Objetos.
- Clases e instancias.
- Encapsulamiento.
- Jerarquías de clase.
- Herencia. Polimorfismo.
- Lenguajes y aplicaciones.

PROGRAMA ANALÍTICO

Unidad 1: La crisis del software. Problemas de las técnicas tradicionales (procedurales). Resolución de problemas complejos. El problema de la extensibilidad, el reuso y el mantenimiento.

Unidad 2: Conceptos básicos: Tipos Abstractos de Datos. Encapsulamiento. Information hiding. Objetos y Programa. Comportamiento de un Objeto. Mensaje y Método. Clasificación: Clases e Instancias. Instanciación. Jerarquías de Clases. Relación isA. Generalización/Especialización. Herencia, Herencia Simple. Clases Abstractas. Hacia mayor genericidad de código: polimorfismo y binding dinámico.

Unidad 3: Relaciones entre Objetos. Relación de conocimiento. Relación isPartOf. Conocimiento vs. composición.

Unidad 4: Lenguajes orientados a objetos: variantes. El lenguaje Smalltalk. Tipos de Mensajes. Variables de instancia. PseudoVariables: self y super. Método new. Biblioteca de clases, jerarquías predefinidas: clase Magnitude y su protocolo.

Unidad 5: Estructuras de Control: Clases Boolean, False y True. Métodos: or:, and: y not. Definición de bloques de código. Clase Context. Métodos: value y value:. Métodos ifTrue:, ifFalse:, ifTrue: ifFalse:, whileTrue:, whileFalse:.

Unidad 6: Estructuras de datos como Objetos. Objetos contenedores. Colecciones de Objetos. Clase Collection y sus subclases Array, OrderedCollection, Set, Dictionary y SortedCollection. Protocolo estándar. Iteradores: to: do:, to: by: do:, timesRepeat:. El iterador do:. Otros iteradores: select:, detect:, reject:, collect:, inject: to:.

Unidad 7: Introducción al lenguaje de Modelado Unificado (Unified Modeling Language). Diagramas de UML. Diagramas de Estructura Estática: Diagramas de Clases. Diagramas Dinámicos ó de Comportamiento: Diagramas de Interacción (Diagramas de Secuencia y Diagramas de Colaboración), Diagramas de Casos de Uso.

Unidad 8: Diseños complejos: uso de self y super combinados. Herencia vs. composición. Doble dispatching.

BIBLIOGRAFÏA

Bibliografía Obligatoria

The Object-Oriented Thought Process, Matt Weisfeld, Third Edition, Pearson Education, Addison Wesley. ISBN-13: 978-0-672-33016-2

Introduction to Object-Oriented Programming, An (3rd Edition), Thimoty Budd,Addison Wesley; 3 edition (2001), ISBN-10: 0201760312

Joy of Smalltalk. Ivan TOMEK, http://plato.acadiau.ca/courses/comp/tomek/jos.htm

Pharo by example. 2009. Andrew P. Black, Stéphane Ducasse, Oscar Nierstrasz and Damien Pollet . Published by Square Bracket Associates, Switzerland. ISBN: 978-3-9523341-4-0. http://pharobyexample.org/versions/PBE1-2009-10-28.pdf http://pharobyexample.org/es/PBE1-sp.pdf

UML GOTA A GOTA. FOWLER MARTIN, SCOTT KENDALL, ADDISON-WESLEY IBEROA. Edición 1999 ISBN 9684443641

Bibliografía complementaria

Designing Object-Oriented Software. Rebecca Wirfs-Brock, Brian Wilkerson (Contributor), Lauren Wiener. Prentice Hall PTR; (January 1991), ISBN 0136298257

ANALISIS Y DISEÑO ORIENTADO A OBJETOS CON APLICACIONES. BOOCH GRADY, ADDISON-WESLEY IBEROA, Edición 1996, ISBN 9684443528

Smalltalk: An Introduction to Application Development Using VisualWorks. Trevor Hopkins, Bernard Horan, Prentice Hall, ISBN: 0133183874

Smalltalk With Style. Suzanne Skublics, Edward J. Klimas, David A. Thomas, John Pugh (Foreword). Pearson Education POD; 1 edition (May 21, 2002), ISBN: 0131655493

Smalltalk Best Practice Patterns. Kent Beck, Prentice Hall PTR; 1st edition (October 3, 1996) ISBN: 013476904X

EL LENGUAJE UNIFICADO DE MODELADO . BOOCH GRADY, JACOBSON IVAR , RUMBAUGH JAMES, ADDISON-WESLEY IBEROA, Edición 2000, ISBN 8478290281

METODOLOGÍA DE ENSEÑANZA

Expectativas de logro

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

Existen tres objetivos particulares:

Escribir Programas Orientados a Objetos

Este objetivo implica:

- adquirir los conocimientos teóricos básicos de la Programación Orientada a Objetos: Tipos Abstractos de Datos. Encapsulamiento. Information hiding. Objetos y Programa Orientado a Objetos. Comportamiento de un Objeto. Mensaje y Método. Clasificación: Clases e Instancias. Instanciación. Jerarquías de Clases. Relación *isA. G*eneralización/Especialización. Herencia, Herencia Simple. Clases Abstractas. Hacia mayor genericidad de código: polimorfismo y binding dinámico.
- experimentar con los conceptos teóricos en situaciones prácticas donde manifiesten las ventajas del paradigma orientado a objetos:
- utilizar un lenguaje orientado a objetos que permita codificar programas orientados a objetos. El aprendizaje de un lenguaje debe incluir la definición d clases, variables, Instanciación de objetos, mecanismos de herencia soportado por el lenguaje; biblioteca de clases. Se sugieren como lenguaje Smalltalk, Ruby o Java.
- familiarización con un ambiente de desarrollo orientado a objetos, como puede ser VisualWorks en el caso del lenguaje Smalltalk

Manejo Adecuado de una Notación

Es necesario contar con alguna notación que facilite la comunicación, documentación y desarrollo de un diseño orientado a objetos. Se introducirá también el uso de un lenguaje de modelado gráfico orientado a objetos (UML), que le permitirá construir diagramas especificando distintos aspectos de un sistema.

Esta notación también debe acompañar el proceso de desarrollo de sistemas orientado a objeto como es el proceso de desarrollo unificado basado en UML (RUP).

Dentro de este objetivo también se encuentra la familiarización con un ambiente que soporte el diseño de diagramas UML e incluso el proceso de desarrollo. Se recomienda el uso de ambientes del estilo de Rational Rose.

Procedimientos didácticos

En función de los objetivos planteados se propone organizar el dictado de la materia en clases teórico-practicas organizadas de la siguiente manera.

Clases Teóricas:

Las clases teóricas están destinadas a presentar los conceptos teóricos del programa de la materia. Los conceptos teóricos deben ser presentados a través de su motivación, su definición, relación e interacción

con los demás conceptos. Estas actividades deben ser fuertemente soportadas por el uso de ejemplos concretos.

Las clases teóricas deben ser presenciales. El docente a cargo debe presentar el tema del día a través de la exposición oral del mismo, promoviendo la participación de los alumnos. La participación del los alumnos se logra a través de la discusión de situaciones concretas de aplicación de los conceptos teóricos en cuestión. Es responsabilidad del docente tener la habilidad de manejar dichas discusiones de manera que se planteen pros y contras de los distintos enfoque expuestos por los alumnos.

Clase practicas

Las clases prácticas deben ser dedicadas a aplicar los conceptos teóricos impartidos en las clases teóricas. Las mismas deben ser guiadas por un trabajo práctico. Cada trabajo practico debe identificar una temática y un conjunto de objetivos teóricos-prácticos a lograr con las ejercitaciones planteadas.

El desarrollo de la clase práctica debe contar con una explicación del trabajo práctico por el auxiliar docente a cargo, donde se le indiquen al alumno los objetivos de la práctica y los conceptos teóricos que se pretenden aplicar, más un conjunto de guías para la resolución de los problemas planteados. Luego, los alumnos desarrollan la práctica llevando a cabo cada ejercitación y contando permanentemente con la posibilidad de trabajar en conjunto con un auxiliar docente que guíe su trabajo y evacue sus dudas.

En la cátedra se plantean actividades planificadas para los alumnos y se les propone resolverlas con las herramientas que cuentan al llegar a la materia (por ejemplo, lenguajes y metodologías no-orientadas a objetos a objetos). Se los “desafía” a presentar la posible evolución de la solución para ese tipo de problema y en podría mejorarse la solución/soluciones actuales - dicha actividad genera el contexto adecuado para luego introducir las propuestas de la POO a dichos desafíos (y así buscar un aprendizaje significativo)

Si bien el curso se apoya en una tecnología en particular (Smalltalk), se mencionan otras alternativas (algunas cubiertas en otras asignaturas, otras que los alumnos pueden conocer de fuentes on-line) y se contrasta con ellas. Esto lleva al alumno a buscar bibliografía relacionada con los cambios tecnológicos relativos a la POO y formarse un criterio sobre las tendencias (por ejemplo, los lenguajes dinámicos, la programación con prototipos, etc.). La cátedra acompaña al alumno en el proceso de interpretación evolutiva de la disciplina, para contrastar sus conclusiones y validar su habilidad para esta competencia.

Los ejercicios de los trabajos prácticos serán diseñados de manera que los mismo impliquen el diseño de un programa, su implementación en un lenguaje OO y su testeo a través de usar tecnicas de testeo de unidad. En la cátedra se pone énfasis en el proceso de identificación de problemas del mundo real, especificación de los mismos como problemas resolubles desde la informática y en el desarrollo de soluciones verificables para los mismos. Se tratan proyectos multidisciplinarios (los requerimientos y dominios de aplicación tomados como ejemplo provienen de muchas disciplinas). Además de ofrecer ejemplos motivadores que sirvan como práctica de las temáticas de la asignatura, se trata de acompañar al alumno (aprovechando la experiencia de trabajo multidisciplinar del equipo docente) en la interpretación del rol del Informático como articulador de soluciones en áreas de conocimiento muy diferentes que requieren participación de expertos extra-disciplinares. Para la especificación y el diseño del programa se usarán los recursos del lenguaje de modelado UML. UML es un lenguaje gráfico con sintaxis formalmente definida, a través de la técnica del metamodelado. Esto permite contar con una especificación precisa del sistema bajo desarrollo. En cuanto a la implementación se sugiere trabajar con el lenguaje Smalltalk y el ambiente Pharo. Para el testeo, cada práctica será acompañada por un test de unidad.

Cada trabajo práctico debe concluir con la entrega para su visado de un ejercicio de manera que el alumno reciba de parte de los docentes comentarios que permitan retroalimentar su evolución en el proceso de aprendizaje. Un trabajo integrador (y otras entregas grupales) ponen en juego y fortalecen la capacidad del

alumno para gestionar, planificar, ejecutar y controlar proyectos de Informática (a pequeña escala). El grado con el que alcanza estas competencias se refleja directamente en el resultado del proyecto integrador y demás entregas.

De acuerdo a la temática desarrollada por cada trabajo práctico las clases prácticas se pueden desarrollar en máquina o no. Los prácticos en máquina tienen como objetivo desarrollar las habilidades de programación en un ambiente de desarrollo asistida por el auxiliar docente. Las prácticas que no son en máquina apuntan a desarrollar diseños los cuales son supervisados por el auxiliar docente.

En cuanto a las interfaces humano computador, en la materia se hace una primera aproximación a la construcción de aplicaciones interactivas orientadas a objetos (que se reforzará en Orientación a Objetos 2), haciendo énfasis en la importancia de la separación modelo-vista, característica fundamental para poder incorporar las competencias de diseño de interacción sin que se vean atadas a la evolución de la lógica del dominio.

EVALUACIÓN

Los alumnos deben aprobar los trabajos prácticos previo a someterse a un examen final para la aprobación de la materia.

Para la aprobación de los trabajos prácticos los alumnos se someten a un examen teórico-práctico al final del curso donde se evalúan los conceptos teóricos-prácticos enunciados en los contenidos de la materia. La modalidad de este examen puede ser escrita, en máquina o ambas dependiendo de la disponibilidad de laboratorio y/o docentes. El mismo contará con al menos una posibilidad de recuperación.

Con respecto a la aprobación de la materia los alumnos pueden elegir entre dos modalidades de evaluación: rendir un examen final teórico-práctico globalizador o realizar un proyecto final. El proyecto consiste en el desarrollo de un sistema de software usando la tecnología de objetos a través del cual se evalúan los conocimientos teórico-prácticos adquiridos. El trabajo final debe ser defendido por el alumno en una exposición oral.

La evaluación de las competencias generales y específicas a cuya adquisición esta cátedra contribuye, tiene lugar en el examen parcial y las múltiples entrevistas y discusiones que el alumno tiene con los docentes durante el desarrollo de los trabajos prácticos visables y los trabajos/proyectos integradores (a entregar), y se refleja en la calificación de los mismos.

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Fecha	Contenidos/Actividades
1	Semana del 26/08/19	Introducción a la POO. Objetos y Clases, Mensajes y Métodos. Prácticas en Smalltalk. Objetos que conocen a otros. Pseudovariable self/this
2	Semana del 02/09/19	El sistema orientado a objetos UI/Modelo - Modularización/Cohesión/Acoplamiento. Casos de uso. Ejemplo conductor de sistema mediano. Las colecciones como objetos. Prácticas en Smalltalk.
3	Semana del 09/09/19	Casos de uso (continuación). Ejemplo conductor de sistema mediano. Las colecciones como objetos. Prácticas en Smalltalk.
4	Semana del 16/09/19	Delegación. Encapsulamiento. Binding dinámico. Polimorfismo. Prácticas en Smalltalk..
6	Semana del 23/09/19	Herencia. Pseudovariable super. Prácticas en Smalltalk.
7	Semana del 30/09/19	Tests de unidad – Como escribir buenos tests. Prácticas en Smalltal con SUnit.
8	Semana del 07/10/19	UML (diagramas de clase y de secuencia). Prácticas en papel y con editores.
9	Semana del 14/10/19	Identificar clases y responsabilidades. Heurísticas de diseño.
10	Semana del 21/10/19	Identificar clases y responsabilidades (continuación). Herencia y/o composición. Parcialito de promoción
11	Semana del 28/10/19	Diseño orientado a objetos (caso de estudio)
12	Semana del 04/11/19	Diseño orientado a objetos (caso de estudio)
13	Semana del 11/11/19	Aplicaciones interactivas orientadas a objetos
14	Semana del 18/11/19	Resolución conjunta de ejercicios prácticos de diseño e implementación. Evaluación.
15	Semana del 25/11/19	Resolución conjunta de ejercicios prácticos de diseño e implementación
16	Semana del 02/12/19	Consultas. Evaluación (recuperatorio)
17	Semana del 09/12/19	Consultas.
18	Semana del 16/12/19	Evaluación (recuperatorio)

Evaluaciones previstas	Fecha
Parcialito de promoción	Semana del 21/10
Parcial	Semana del 11/11
Recuperatorio	Semana del 2/12
Recuperatorio	Semana del 16/12

Contacto de la cátedra (mail, sitio WEB, plataforma virtual de gestión de cursos):

alicia.diaz@lifia.info.unlp.edu.ar
gustavo.rossi@lifia.info.unlp.edu.ar
roxana.giandini@lifia.info.unlp.edu.ar

Firma del/los profesor/es

Orientación a Objetos 1 Redictado	**Carrera/ Plan:** *(Dejar lo que corresponda)* *Licenciatura en Informática* Plan 2015/Plan 2012/Plan 2003-07 *Licenciatura en Sistemas* Plan 2015/Plan 2012/Plan 2003-07 *Analista Programador Universitario* Plan 2015/Plan 2007 *Analista en TIC* Plan 2017
Año 2019	**Año:** 2019 **Régimen de Cursada:** *Semestral* **Carácter (Obligatoria/Optativa):** Obligatoria **Correlativas**: Taller de Programación **Profesor/es:** Gustavo Rossi, Andrés Rodriguez, Matías Urbieta **Hs. semanales**: 6 hrs

FUNDAMENTACIÓN

La Programación Orientada a Objetos (POO) es fundamental en la formación del profesional informático. La misma aporta técnicas de programación que combinan la abstracción, modularización, encapsulamiento junto con el polimorfismo y herencia, promoviendo así una forma de programación basada en la descomposición adecuada en objetos de manera de favorecer la modificabilidad, mantenimiento y reuso. Aspectos que son rectores en el desarrollo de un programa OO.

OBJETIVOS GENERALES

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

COMPETENCIAS

- CGS6- Capacidad para interpretar la evolución de la Informática con una visión de las tendencias tecnológicas futuras.
- CGT1- Identificar, formular y resolver problemas de Informática.
- CGT2- Concebir, diseñar y desarrollar proyectos de Informática.
- CGT5- Utilizar de manera efectiva las técnicas y herramientas de aplicación de la Informática.
- CGT8 Capacidad de interpretación y resolución de problemas multidisciplinarios, desde los conocimientos de la disciplina informática.
- LI - CE4 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real, especificación formal de los mismos, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software/sistemas de información que se ejecuten sobre equipos de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.
- LS - CE1 – Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real. Especificación formal, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software que se ejecuten sobre sistemas de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.

CONTENIDOS MINIMOS (de acuerdo al Plan de Estudios)

- Objetos.
- Clases e instancias.
- Encapsulamiento.
- Jerarquías de clase.
- Herencia. Polimorfismo.
- Lenguajes y aplicaciones.

PROGRAMA ANALÍTICO

Unidad 1: La crisis del software. Problemas de las técnicas tradicionales (procedurales). Resolución de problemas complejos. El problema de la extensibilidad, el reuso y el mantenimiento.

Unidad 2: Conceptos básicos: Tipos Abstractos de Datos. Encapsulamiento. Information hiding. Objetos y Programa. Comportamiento de un Objeto. Mensaje y Método. Clasificación: Clases e Instancias. Instanciación. Jerarquías de Clases. Relación isA. Generalización/Especialización. Herencia, Herencia Simple. Clases Abstractas. Hacia mayor genericidad de código: polimorfismo y binding dinámico.

Unidad 3: Relaciones entre Objetos. Relación de conocimiento. Relación isPartOf. Conocimiento vs. composición.

Unidad 4: Lenguajes orientados a objetos: variantes. El lenguaje Smalltalk. Tipos de Mensajes. Variables de instancia. PseudoVariables: self y super. Método new. Biblioteca de clases, jerarquías predefinidas: clase Magnitude y su protocolo.

Unidad 5: Estructuras de Control: Clases Boolean, False y True. Métodos: or:, and: y not. Definición de bloques de código. Clase Context. Métodos: value y value:. Métodos ifTrue:, ifFalse:, ifTrue: ifFalse:, whileTrue:, whileFalse:.

Unidad 6: Estructuras de datos como Objetos. Objetos contenedores. Colecciones de Objetos. Clase Collection y sus subclases Array, OrderedCollection, Set, Dictionary y SortedCollection. Protocolo estándar. Iteradores: to: do:, to: by: do:, timesRepeat:. El iterador do:. Otros iteradores: select:, detect:, reject:, collect:, inject: to:.

Unidad 7: Introducción al lenguaje de Modelado Unificado (Unified Modeling Language). Diagramas de UML. Diagramas de Estructura Estática: Diagramas de Clases. Diagramas Dinámicos ó de Comportamiento: Diagramas de Interacción (Diagramas de Secuencia y Diagramas de Colaboración), Diagramas de Casos de Uso.

Unidad 8: Diseños complejos: uso de self y super combinados. Herencia vs. composición. Doble dispatching.

BIBLIOGRAFÏA

Bibliografía Obligatoria

The Object-Oriented Thought Process, Matt Weisfeld, Third Edition, Pearson Education, Addison Wesley. ISBN-13: 978-0-672-33016-2

Introduction to Object-Oriented Programming, An (3rd Edition), Thimoty Budd,Addison Wesley; 3 edition (2001), ISBN-10: 0201760312

Joy of Smalltalk. Ivan TOMEK, http://plato.acadiau.ca/courses/comp/tomek/jos.htm

Pharo by example. 2009. Andrew P. Black, Stéphane Ducasse, Oscar Nierstrasz and Damien Pollet . Published by Square Bracket Associates, Switzerland. ISBN: 978-3-9523341-4-0. http://pharobyexample.org/versions/PBE1-2009-10-28.pdf http://pharobyexample.org/es/PBE1-sp.pdf

UML GOTA A GOTA. FOWLER MARTIN, SCOTT KENDALL, ADDISON-WESLEY IBEROA. Edición 1999 ISBN 9684443641

Bibliografía complementaria

Designing Object-Oriented Software. Rebecca Wirfs-Brock, Brian Wilkerson (Contributor), Lauren Wiener. Prentice Hall PTR; (January 1991), ISBN 0136298257

ANALISIS Y DISEÑO ORIENTADO A OBJETOS CON APLICACIONES. BOOCH GRADY, ADDISON-WESLEY IBEROA, Edición 1996, ISBN 9684443528

Smalltalk: An Introduction to Application Development Using VisualWorks. Trevor Hopkins, Bernard Horan, Prentice Hall, ISBN: 0133183874

Smalltalk With Style. Suzanne Skublics, Edward J. Klimas, David A. Thomas, John Pugh (Foreword). Pearson Education POD; 1 edition (May 21, 2002), ISBN: 0131655493

Smalltalk Best Practice Patterns. Kent Beck, Prentice Hall PTR; 1st edition (October 3, 1996) ISBN: 013476904X

EL LENGUAJE UNIFICADO DE MODELADO . BOOCH GRADY, JACOBSON IVAR , RUMBAUGH JAMES, ADDISON-WESLEY IBEROA, Edición 2000, ISBN 8478290281

METODOLOGÍA DE ENSEÑANZA

Expectativas de logro

Presentar formalmente el paradigma de objetos, sus características, ventajas y aplicaciones dentro del desarrollo de sistemas de software. Desarrollar prácticas concretas con lenguajes orientados a Objetos. Establecer metodologías de análisis y diseño orientados a objetos.

Existen tres objetivos particulares:

Escribir Programas Orientados a Objetos

Este objetivo implica:

- adquirir los conocimientos teóricos básicos de la Programación Orientada a Objetos: Tipos Abstractos de Datos. Encapsulamiento. Information hiding. Objetos y Programa Orientado a Objetos. Comportamiento de un Objeto. Mensaje y Método. Clasificación: Clases e Instancias. Instanciación. Jerarquías de Clases. Relación *isA. G*eneralización/Especialización. Herencia, Herencia Simple. Clases Abstractas. Hacia mayor genericidad de código: polimorfismo y binding dinámico.
- experimentar con los conceptos teóricos en situaciones prácticas donde manifiesten las ventajas del paradigma orientado a objetos:
- utilizar un lenguaje orientado a objetos que permita codificar programas orientados a objetos. El aprendizaje de un lenguaje debe incluir la definición d clases, variables, Instanciación de objetos, mecanismos de herencia soportado por el lenguaje; biblioteca de clases. Se sugieren como lenguaje Smalltalk, Ruby o Java.
- familiarización con un ambiente de desarrollo orientado a objetos, como puede ser VisualWorks en el caso del lenguaje Smalltalk

Manejo Adecuado de una Notación

Es necesario contar con alguna notación que facilite la comunicación, documentación y desarrollo de un diseño orientado a objetos. Se introducirá también el uso de un lenguaje de modelado gráfico orientado a objetos (UML), que le permitirá construir diagramas especificando distintos aspectos de un sistema.

Esta notación también debe acompañar el proceso de desarrollo de sistemas orientado a objeto como es el proceso de desarrollo unificado basado en UML (RUP).

Dentro de este objetivo también se encuentra la familiarización con un ambiente que soporte el diseño de diagramas UML e incluso el proceso de desarrollo. Se recomienda el uso de ambientes del estilo de Rational Rose.

Procedimientos didácticos

El curso está dirigido a alumnos que previamente cursaron Orientación a Objetos 1, por lo cual, durante el redictado, se hará más énfasis en las actividades prácticas, repasando y reforzando los conceptos teóricos. En función de los objetivos planteados se propone organizar el dictado de la materia en clases teórico-prácticas organizadas de la siguiente manera.

Habrá una clase semanal (en dos turnos: mañana y tarde), en donde se presentaran ejercicios con el fin de repasar los conceptos teóricos necesarios para resolverlos. Luego del repaso, los alumnos deberán hacer consultas sobre los ejercicios de las actividades prácticas para poder resolverlos.

Se dispondrá de una plataforma para que los alumnos puedan estar en contacto y obtener respuesta de los docentes en todo momento.

Las clases son de modalidad presencial. El docente a cargo debe presentar el tema del día a través de la exposición oral del mismo, promoviendo la participación de los alumnos. La participación del los alumnos se logra a través de la discusión de situaciones concretas de aplicación de los conceptos teóricos en cuestión. Es responsabilidad del docente tener la habilidad de manejar dichas discusiones de manera que se planteen pros y contras de los distintos enfoque expuestos por los alumnos.

Las actividades prácticas deben ser dedicadas a aplicar los conceptos a problemáticas concretas. Las mismas deben ser guiadas por un trabajo práctico. Cada trabajo practico debe identificar una temática y un conjunto de objetivos teóricos-prácticos a lograr con las ejercitaciones planteadas.

El desarrollo de la actividad práctica debe contar con una explicación del trabajo práctico por el auxiliar docente a cargo, donde se le indiquen al alumno los objetivos de la práctica y los conceptos teóricos que se pretenden aplicar, más un conjunto de guías para la resolución de los problemas planteados. Luego, los alumnos desarrollan la práctica llevando a cabo cada ejercitación y contando permanentemente con la posibilidad de trabajar en conjunto con un auxiliar docente que guíe su trabajo y evacue sus dudas.

En la cátedra se plantean actividades planificadas para los alumnos y se les propone resolverlas con las herramientas que cuentan al llegar a la materia (por ejemplo, lenguajes y metodologías no-orientadas a objetos a objetos). Se los “desafía” a presentar la posible evolución de la solución para ese tipo de problema y en podría mejorarse la solución/soluciones actuales - dicha actividad genera el contexto adecuado para luego introducir las propuestas de la POO a dichos desafíos (y así buscar un aprendizaje significativo)

Si bien el curso se apoya en una tecnología en particular (Smalltalk), se mencionan otras alternativas (algunas cubiertas en otras asignaturas, otras que los alumnos pueden conocer de fuentes on-line) y se contrasta con ellas. Esto lleva al alumno a buscar bibliografía relacionada con los cambios tecnológicos relativos a la POO y formarse un criterio sobre las tendencias (por ejemplo, los lenguajes dinámicos, la programación con prototipos, etc.). La cátedra acompaña al alumno en el proceso de interpretación evolutiva de la disciplina, para contrastar sus conclusiones y validar su habilidad para esta competencia.

Los ejercicios de los trabajos prácticos serán diseñados de manera que los mismo impliquen el diseño de un programa, su implementación en un lenguaje OO y su testeo a través de usar tecnicas de testeo de unidad. En la cátedra se pone énfasis en el proceso de identificación de problemas del mundo real, especificación de los mismos como problemas resolubles desde la informática y en el desarrollo de soluciones verificables para los mismos. Se tratan proyectos multidisciplinarios (los requerimientos y dominios de aplicación tomados como ejemplo provienen de muchas disciplinas). Además de ofrecer ejemplos motivadores que sirvan como práctica de las temáticas de la asignatura, se trata de acompañar al alumno (aprovechando la experiencia de trabajo multidisciplinar del equipo docente) en la interpretación del rol del Informático como articulador de soluciones en áreas de conocimiento muy diferentes que requieren participación de expertos extra-disciplinares. Para la especificación y el diseño del programa se usarán los recursos del lenguaje de modelado UML. UML es un lenguaje gráfico con sintaxis formalmente definida, a través de la técnica del metamodelado. Esto permite contar con una especificación precisa del sistema bajo desarrollo. En cuanto a la implementación se sugiere trabajar con el lenguaje Smalltalk y el ambiente Pharo. Para el testeo, cada práctica será acompañada por un test de unidad.

Cada trabajo práctico debe concluir con la entrega para su visado de un ejercicio de manera que el alumno reciba de parte de los docentes comentarios que permitan retroalimentar su evolución en el proceso de aprendizaje. Un trabajo integrador (y otras entregas grupales) ponen en juego y fortalecen la capacidad del alumno para gestionar, planificar, ejecutar y controlar proyectos de Informática (a pequeña escala). El grado con el que alcanza estas competencias se refleja directamente en el resultado del proyecto integrador y demás entregas.

De acuerdo a la temática desarrollada por cada trabajo práctico las clases prácticas se pueden desarrollar en máquina o no. Los prácticos en máquina tienen como objetivo desarrollar las habilidades de programación en un ambiente de desarrollo asistida por el auxiliar docente. Las prácticas que no son en máquina apuntan a desarrollar diseños los cuales son supervisados por el auxiliar docente.

En cuanto a las interfaces humano computador, en la materia se hace una primera aproximación a la construcción de aplicaciones interactivas orientadas a objetos (que se reforzará en Orientación a Objetos 2), haciendo énfasis en la importancia de la separación modelo-vista, característica fundamental para poder incorporar las competencias de diseño de interacción sin que se vean atadas a la evolución de la lógica del dominio.

EVALUACIÓN

Los alumnos deben aprobar los trabajos prácticos previo a someterse a un examen final para la aprobación de la materia.

Para la aprobación de los trabajos prácticos los alumnos se someten a un examen teórico-práctico al final del curso donde se evalúan los conceptos teóricos-prácticos enunciados en los contenidos de la materia. La modalidad de este examen puede ser escrita, en máquina o ambas dependiendo de la disponibilidad de laboratorio y/o docentes. El mismo contará con al menos una posibilidad de recuperación.

Con respecto a la aprobación de la materia los alumnos pueden elegir entre dos modalidades de evaluación: rendir un examen final teórico-práctico globalizador o realizar un proyecto final. El proyecto consiste en el desarrollo de un sistema de software usando la tecnología de objetos a través del cual se evalúan los conocimientos teórico-prácticos adquiridos. El trabajo final debe ser defendido por el alumno en una exposición oral.

La evaluación de las competencias generales y específicas a cuya adquisición esta cátedra contribuye, tiene lugar en el examen parcial y las múltiples entrevistas y discusiones que el alumno tiene con los docentes durante el desarrollo de los trabajos prácticos visables y los trabajos/proyectos integradores (a entregar), y se refleja en la calificación de los mismos.

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Fecha	Contenidos/Actividades
1	Semana del 18/03/19	Introducción a la POO. Objetos y Clases, Mensajes y Métodos. Prácticas en Smalltalk. Objetos que conocen a otros. Pseudovariable self/this
2	Semana del 25/03/19	El sistema orientado a objetos UI/Modelo - Modularización/Cohesión/Acoplamiento. Casos de uso. Ejemplo conductor de sistema mediano. Las colecciones como objetos. Prácticas en Smalltalk.
3	Semana del 01/04/19	Casos de uso (continuación). Ejemplo conductor de sistema mediano. Las colecciones como objetos. Prácticas en Smalltalk.
4	Semana del 08/04/19	Delegación. Encapsulamiento. Binding dinámico. Polimorfismo. Prácticas en Smalltalk..
6	Semana del 15/04/19	Herencia. Pseudovariable super. Prácticas en Smalltalk.
7	Semana del 22/04/19	Tests de unidad – Como escribir buenos tests. Prácticas en Smalltal con SUnit.
8	Semana del 29/04/19	UML (diagramas de clase y de secuencia). Prácticas en papel y con editores.
9	Semana del 06/05/19	Identificar clases y responsabilidades. Heurísticas de diseño.
10	Semana del 13/05/19	Identificar clases y responsabilidades (continuación). Herencia y/o composición.
11	Semana del 20/05/19	Diseño orientado a objetos (caso de estudio)
12	Semana del 27/05/19	Diseño orientado a objetos (caso de estudio)
13	Semana del 03/06/19	Aplicaciones interactivas orientadas a objetos
14	Semana del 10/06/19	Resolución conjunta de ejercicios prácticos de diseño e implementación. Evaluación.
15	Semana del 17/06/19	Resolución conjunta de ejercicios prácticos de diseño e implementación
16	Semana del 24/06/19	Consultas. Evaluación (recuperatorio)
17	Semana del 01/07/19	Consultas.
18	Semana del 08/07/19	Evaluación (recuperatorio)

Evaluaciones previstas	Fecha
Parcial	Semana del 10/06

Recuperatorio	Semana del 24/6
Recuperatorio	Semana del 8/7

Contacto de la cátedra (mail, sitio WEB, plataforma virtual de gestión de cursos):

Gustavo@lifia.info.unlp.edu.ar
Andres@lifia.info.unlp.edu.ar
Matias.urbieta@lifia.info.unlp.edu.ar

Firma del/los profesor/es